

Modeling and Manipulation of Amorphous Objects

Xiaoming Wei
Center for Visual Computing (CVC)
and Department of Computer Science
State University of New York at Stony Brook
Research Proficiency Exam
Advisor: Prof. Arie Kaufman

January 23 2001

Contents

0.1	Introduction	1
0.1.1	Volume Rendering	1
0.1.2	Applications	2
0.1.3	Design Goals	2
0.2	Procedural Modeling Techniques	3
0.2.1	Solid Space	3
0.2.2	Particle System Modeling	4
0.3	Physically Based Modeling	6
0.3.1	Background	6
0.3.2	Navier-Stokes Equation	6
0.3.3	Advection-Diffusion Equation	14
0.3.4	Turbulence Theory	15
0.4	Cellular Automata	16
0.4.1	Background	16
0.4.2	Lattice Gas Model	16
0.4.3	Lattice Boltzman Model	18
0.4.4	Parallel Optimization	20
0.5	Conclusion	20

0.1 Introduction

Clouds forming, disappearing and moving across the sky, curling wisps of smoke, mist blowing across a field, car exhaust, steam rising from a coffee mug, a swirl of dust in the room, etc., all these are realistic portrayals of gaseous natural phenomena. Amorphous objects, such as clouds, dust, smoke, haze, fog, gas and fire, need to be implemented in order to have a convincing realistic environment. The problem of how to model the objects and their interaction with the surrounding environment has been an active area of research in graphics literature. The conventional method is to use a surface modeling approach. Amorphous and translucent objects can also be represented by surface methods. However, these methods can't efficiently support the modeling and rendering of amorphous phenomena that are volumetric in nature and lack any tangible surfaces.

Clouds are phenomena that we see every day. They play an important role in the animation of flight simulation and outdoor scenes. They change in shape and color all the time. Before thinking about the modeling method of clouds, let's understand the physical process of cloud formation first. The whole process can be outlined as follows. In the beginning, clouds are formed as bubbles of air. The air is heated by the underlying terrain heat, causing the bubble to become less dense, and to rise into regions of lower pressure in which the bubble expands. Expansion cools the bubble, increasing the relative humidity inside. Then, water vapor in the bubble becomes water droplets. Clouds are formed. Based on their appearance, we can divide clouds into three basic classes: cirrus, stratus and cumulus. Cirrus clouds are wispy clouds at high altitudes. Stratus clouds are layer clouds with no distinct detail, lying at a low altitude. Cumulus clouds are heap clouds at low altitudes.

Fires are turbulent, non-steady objects. They result from the combustion of fuels and oxidizers. As the molecules of these compounds meet at a sufficiently high temperature, a chemical reaction becomes possible. The result of burning components are called the fires. Smoke will be generated because of incomplete combustion at a lower temperature in the fire. In animation related to smoke, soot accumulation and scorching over time should also be modeled to describe surrounding objects. Dust is generated when a moving vehicle or animal passes on an unpaved road. Its property depends on many factors, such as the condition of the road, and the speed and weight of the object. There are also many other amorphous objects, such as gas, steam, haze and fog, etc. Please refer to the physics literature for the exact description of their physical processes.

0.1.1 Volume Rendering

The *volume rendering* approach is a basic method that can be used to visualize amorphous objects. Over the years, many techniques have been developed to visualize volumetric data. Most of the early methods involved approximating a surface contained within the data using geometric primitives. When such surface rendering is used, a dimension of information is lost, and adequate approximations may result in an excessive number of primitives. Also, the amorphous phenomena can't be represented adequately using surfaces based on their dynamic behavior and inherent volumetric data properties. Because of these problems, the *volume rendering* techniques have been developed, attempting to capture the entire 3D data in a single 2D image by projecting a 2D image directly from the 3D volumetric data.

Volume rendering [5, 26] can be divided into three main approaches: object-order, image-order and domain methods. In object-order methods, the contribution of each voxel to the screen pixels is calculated and the combined contribution yields the final image. One such method is called *splatting*. In this method, the volume is thought of as a field of overlapping interpolation kernel h . One such kernel is placed at each voxel location and weighted by the voxel's intensity value. The overlapping voxel kernels then reconstruct a continuous representation of the volume. In the image-order method such as *ray casting*, we cast viewing rays into the volume and calculate the integration by sampling the volume along the ray and compositing the samples in front-to-back or back-to-front order. In domain methods, the spatial data is transformed into an alternative domain, such as a compression, wavelet or frequency domain, from which a projection is directly generated. Volume rendering is a well-developed area, please refer to Kaufman's book [24, 25] for further details.

0.1.2 Applications

Clouds are a critical element in flight simulation and air-to-air combat. They are also important in the simulation of intelligent weapon systems, which seek and identify aerial targets in cluttered backgrounds. Realistic cloud simulation would also be an effective tool in the field of meteorology. Since clouds are such familiar objects in everyday life, it is desirable to simulate them effectively for other applications, such as entertainment, advertising, training and the arts.

Graphics simulations of fluid behavior are also in great demand in film making for depicting gases, liquids, smoke, dust, fire and other amorphous phenomena. In traditional animation making, there is no simulation at all. Artists need to draw every frame of the animation. This approach gives a wide range of flexibility and control. However, it is a tedious process with realistic limits on the complexity that can be achieved. A system with good amorphous object models can not only support simple user set-ups, but also describe physically accurate object movements, generating lots of high quality animation, rich in complexity and guaranteed realistic motion. Paint programs can also benefit from fluid solvers to emulate traditional techniques such as water color and oil paint. Another possible application is to use turbulent flow method in texture synthesis. Environmental research is another application of the amorphous object modeling. People have used it in the study of pool fire and volcanic ash clouds [37, 38].

0.1.3 Design Goals

To achieve realism, the methods we use should at least satisfy the following conditions:

1. The method should support real-time applications. Speed is always important in most simulation systems.
2. The model should be general enough to represent a variety of phenomena.
3. Other objects in the environment must be able to interact with the effect. In order to make the scene real, amorphous objects should interact with other objects in a physically correct manner, such as moving along the wind direction.
4. The model should reside in the three-dimensional space.

0.2 Procedural Modeling Techniques

0.2.1 Solid Space

Solid spaces [11] are three-dimensional spaces associated with an object that allows for control of an attribute of the object. It encompasses traditional solid texturing, hypertextures, and volume density functions. By changing the solid space over time or moving points through the solid space, we can get animation.

Solid texture was introduced by Perlin [33]. It is generated by evaluating functions at the visible surface points of the object. *Noise()* is the most fundamental stochastic functions in the solid texture approach. It is defined as a scalar valued function which takes a three dimensional vector as its argument. It also has the properties of statistical invariance under rotation, narrow bandpass limit in frequency and the statistical invariance under translation. The function *turbulence()* can be defined based on *Noise()*:

```
Function turbulence(p)
  t=0
  scale=1
  while(scale > pixelsize)
    t +=abs(Noise(p/scale)*scale)
    scale /=2
  return t
```

The inner loop of this function indicates that the amount of *Noise()* added is proportional to the scale size. In this way, we can obtain a self-similar pattern of perturbation. Also, while the deformation is continuous everywhere, the *abs()* at each iteration assures that its gradient will have discontinuous boundaries at all scales. This will give a visual impression of discontinuous flow, which will be interpreted by the viewer as turbulent. To simulate gaseous phenomena, solid texture can be used to control the transparency of objects that define the space the gaseous substance occupies.

Ebert [12] used a volume density function to represent clouds. He proposed to combine the A-buffer and volume rendering techniques to render a scene with surface modeled objects and volumetric data.

Hypertexture [34] is formed by combining solid texture with a function defining the fuzzy(soft) region of objects. This approach is essentially an extension of procedural solid texture. It evaluated throughout a volumetric region instead of only at surfaces. By defining a function $D(x)$ that maps points in 3D space to floating numbers between 0 and 1, we can obtain the fuzzy region of an object. This is also a good amorphous object modeling method. For example, a fire ball can be obtained by using the density equation $D(x)=\text{sphere}(x(1+\text{turbulence}(x)))$. The color map is structured in this case so that low densities map to red, high densities to yellow.

Another procedural modeling method is to use fractals [30]. A *fractal* is defined as a geometrically complex object, the complexity of which arises through the repetition of form over some range of size. There are at least two kinds of complexity in the world: fractal and nonfractal. Nonfractal complexity is characterized by the accumulation of a variety of features through distinct and unrelated events over time. Most natural phenomena, such as turbulent flows are fractal. They have statistical self-similarity, which means the smaller part of the object looks just like the larger part of the object.

Early in 1985, Gardner [20] described an approach using plane and ellipsoids to model 2D and 3D clouds. He also used a mathematical texturing function composed of short sums of sine waves(Fourier synthesis) to decide the shading and translucence of the surfaces. His approach can model the horizontal cloud formation(cloud layers) or vertical development(cumuliform cloud). Different shapes of clouds can be generated by combining several ellipsoids. As with any other surface-based approach to modeling gases, this method can't produce three-dimensional volumes of gas.

King, et al. [27, 28] proposed a method of using textured splats [7] to achieve fast volume rendering and the animation of amorphous objects. His method is based on splatting. Mostly, people use the gaussian function to stand for the footprint of splats. By combining texture with the gaussian function, they generates a new function which is called textured splats. The difficulty in this method is how to use and get different textures to represent the dynamic behaviors of amorphous phenomena. King et al. suggest using spectral analysis to select the part of the image which contains specific frequency components. The local dynamics of amorphous objects can be generated by getting textures along a certain curve. With specific texture support hardware, good results with real-time speed can be achieved.

0.2.2 Particle System Modeling

The particle system was first introduced by Reeves [35] in 1983. It can represent objects as clouds of primitive particles that occupy their volume, rather than the conventional polygons and patches. A particle system is not a static entity, as the particles can move and change with time. At each time step, new particles are generated into the system; while old particles died from the system. The position, orientation, attributes and dynamics of each particle also change according to a set of stochastic rules. The most commonly used stochastic rule is to define a variable based on user defined mean values and variance values: $N = Mean_N + Rand() \times Variance_N$. If we further define the rules to satisfy partial differential equations, we can take advantage of models that have been developed in other scientific or engineering fields. We will discuss these models later in Section 4.

The particle system is great to model objects that change shape over time. In modeling amorphous objects, it has several advantages over conventional surface-oriented methods:

1. A simple geometric primitive: a particle is much simpler than most other graphical primitives.
2. A procedural model: The model definition is procedural and is controllable by random numbers.

The simulation loop for a particle system can be generalized as follows:

- Create particle systems: ranges of particle activities; data structures to hold particles
- Initialize each particle's parameters(position,velocity, color,transparency and shape, etc.)
- For each simulation frame, do the following:
 - Create and remove particles
 - Calculate and update each particle's parameters

- Render the particles in the particle systems
- Update the particle system’s range of activities

Based on particle systems, we can reduce the complexity of the amorphous objects modeling problem by using discrete particles to represent their gaseous motion. For example, a cloud might be composed of many particle systems, each representing a billowing region of water particles. Simple stochastic processes can be defined to describe the influence on clouds by the wind and terrain. The disadvantages of the particle system approach are the huge amount of particles used to describe complex objects and the difficulty of directly modeling the interaction of fluid with surrounding objects. Figure 1 describes the classification and applications of particle systems.

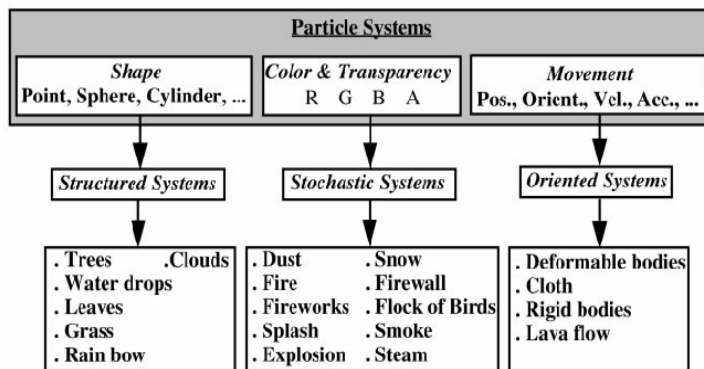


Figure 1: Application of Particle Systems

The particle system was first used to model fire in Paramount’s work [32]. In the film *Star Trek II: The Wrath of Khan*, the wall-of-fire element was generated using a two-level hierarchy of particle systems. The top-level system was centered at the impact point of the genesis bomb. It generated particles which were themselves particle systems. The second-level particle systems were modeled to look like explosions. Each particle system can be seen as a volcano exploding upwards, eventually falling to the planet’s surface due to the pull of gravity, forming a parabolic curve rather than a straight line. Reeves [36] used the structured particle system to model trees and grass. He used a stochastic model to represent gusts of wind and the motion of wind-blown blades of grass. In 1994, Ebert [8] used an inverse particle system to simulate amorphous objects. Each three-dimensional screen space point is inversely mapped back to world space. From world space, it is mapped into the gas and turbulence space through the use of simple affine transformations. Finally, it is moved through the turbulence space over time to create the movement. The main advantage of this approach is that extremely large geometric databases of particles are not required to get realistic images. The complexity is always controlled by the number of screen space points in which the gas is potentially visible. The vector field table is used to allow external programs to generate three dimensional fields that control the gas movement. David also used the functional flow field tables to define for each region of gas which function to evaluate to control its movement. Each flow field table entry can either contain one specific function or a list of functions. For example, the wind effects on gas can be achieved by combining several user-defined attractors. Yaeger and Upson [52] also used the particle system

in their project to generate high resolution images of the planet Jupiter for the movie *2010*. They implemented the flow field of Jupiter by the summation of a one-dimensional shear flow and numerous smaller-scale disturbances.

Because of the parallel nature of particle systems, they are also well suited for highly parallel computation. Sims [43] implemented a general tool for particle system animation and rendering based on a data parallel supercomputer. He also extended Reeve's work and allowed the independent particles to interact with the environment. The system can support both kinematics and dynamic control of the particles.

Chen, et al. [3] combined particle systems, computational fluid dynamics and behavioral simulation techniques to simulate dust behavior generated by a fast moving vehicle in real time. The generation and behavior of dust particles are caused by many factors, such as the natural wind, the speed of the vehicle and the condition of the road. To solve the problem with physical computation would be too time consuming. Thus, Chen used three different particle systems: fluid turbulence, particle momentum and airborne drift together to simulate the dust behavior.

0.3 Physically Based Modeling

0.3.1 Background

To correctly model the interaction between flows and surrounding objects, another approach is to simulate the physical process of fluid dynamics, however, this usually requires a large amount of computation time and memory, especially in 3D. It is almost impossible to get real time speed.

While researchers in the field of Computational Fluid Dynamics (CFD) extensively study the computational model of fluids, their goals are to obtain highly accurate and completely descriptive simulations of fluid behavior. Thus, using their formulas directly will be extremely time consuming. In contrast, the goal of many researchers in the graphics field is to produce realistic-looking fluid movement, interaction, and fast calculation for animation. Thus, most of the current implementation is based on a simplified or special version of fluid equations.

Most of the fluid behavior can be described by the Navier-Stokes equation, arrived at more than 150 years ago by the French engineer, Claude Navier, and the Irish mathematician, George Stokes. In the following subsection, we will give a brief introduction of the Navier-Stokes equation [14].

0.3.2 Navier-Stokes Equation

In computational fluid dynamics, a volume of gas can be represented as a combination of a temperature field, a pressure field and a velocity vector field. What we need to find out is the correlation of these variables. Now we use two dimensions as an example to derive the Navier-Stokes equation. For a two dimensional flow, we will have five flow properties: the two velocity components u, v ; density ρ , temperature T and pressure p . Consider a small control patch, we number each of the surfaces based on Figure 2.

According to the principle of "conservation of mass", the rate at which mass increases within the control patch should equal to the rate at which mass leaves the control patch through its four boundaries. Now the mass within the control patch can be represented by $\rho\Delta x\Delta y$, thus, the time rate of change of mass in the control patch is $\frac{\partial\rho\Delta x\Delta y}{\partial t}$. Next, consider the rate at which mass

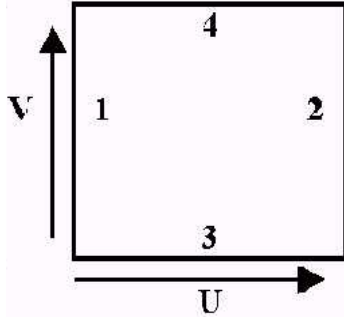


Figure 2: Numbering of the control patch surfaces

enters or leaves through the four boundaries. We use Surface 1 as an example. Since velocity u can be represented by $\frac{dx}{dt}$, the amount of mass entering Surface 1 is $\rho dx \Delta y$. Divided by dt , the rate at which mass enters Surface 1 is $\rho u \Delta y$. The result of the rest of the surfaces can be derived similarly. They are $(\rho u \Delta y)_1, (-\rho u \Delta y)_2, (\rho v \Delta x)_3$ and $(-\rho v \Delta x)_4$. Combining these four terms, we will have

$$\frac{\partial \rho}{\partial t} = \frac{(\rho u)_1 - (\rho u)_2}{\Delta x} + \frac{(\rho v)_3 - (\rho v)_4}{\Delta y} = -\frac{(\rho u)_2 - (\rho u)_1}{\Delta x} - \frac{(\rho v)_4 - (\rho v)_3}{\Delta y} \quad (1)$$

considering Δx and Δy go to zero and based on $\lim_{\Delta x \rightarrow 0} \frac{(\rho u)_2 - (\rho u)_1}{\Delta x} = \frac{\partial \rho u}{\partial x}$, we will get

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0 \quad (2)$$

for incompressible flow, the formula will further reduce to

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (3)$$

Now we think about the conservation of momentum. Before we can proceed any further, we need to have a good understanding of the terms of viscosity, viscous stresses, conductivity, etc. Let's consider the left face of the control patch. The gas molecules to the left of the patch can interact with the patch in one of the following three ways:

- Organized motion from left to right. While the molecules are constantly moving back and forth, over a small period of time, the majority of these molecules either enter the control patch or leave the control patch. This "average" over a period of time is called the *flow velocity*.
- Exchange of u momentum between the molecules on the left and those on the right by collision. In this case, there is no net gain in mass, instead there is a gain (or loss) in momentum. These collision effects may be averaged over a sufficiently small period of time, and may be viewed as a *pressure* force exerted by the fluid on the left face of the control patch.

- Exchange of u and v momentum by random linear motion of molecules jumping in and out of the control patch across the left face. In this case, the molecules will bring the u and v momentum in or out. The time averages of these rates at which the u and v momentum is brought into the control patch across a face are called viscous forces. The forces per unit area are called *viscous stresses*. The viscous stresses that bring u momentum in and out are called *normal viscous stresses*, while those that bring in the v momentum (by entering the face at an angle) are called *tangential viscous stresses*. For solids, when a force or stress is applied, it only undergoes a finite amount of deformation. However, fluid will continuously deform when a shear stress is applied.

We use τ to represent the normal and tangential stresses. They are identified by two subscripts. The first subscript indicates the plane on which they act. The second subscript identifies the direction of the force associated with the face. Now we only consider the *Newtonian fluids*, which satisfy the linear relationship between shear stress and the velocity derivative of the normal of the face. For *Newtonian fluids*, the stresses on each face can be represented by the following Stokes equation:

$$\tau_{xx} = 2\mu \frac{\partial u}{\partial x} - \frac{2}{3}\mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \quad (4)$$

$$\tau_{yy} = 2\mu \frac{\partial v}{\partial y} - \frac{2}{3}\mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \quad (5)$$

$$\tau_{zz} = 2\mu \frac{\partial w}{\partial z} - \frac{2}{3}\mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \quad (6)$$

$$\tau_{xy} = \tau_{yx} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \quad (7)$$

$$\tau_{xz} = \tau_{zx} = \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \quad (8)$$

$$\tau_{zy} = \tau_{yz} = \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \quad (9)$$

where μ is called the molecular viscosity. For air, viscosity will increase with temperature, because viscous effects are associated with random molecular motion. For 2D flow and considering the incompressibility of the flow ($\nabla u = 0$), the Stokes equation will reduce to the four formulas:

$$\tau_{xx} = 2\mu \frac{\partial u}{\partial x} \quad (10)$$

$$\tau_{yy} = 2\mu \frac{\partial v}{\partial y} \quad (11)$$

$$\tau_{xy} = \tau_{yx} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \quad (12)$$

Now we will derive the momentum equation based on Newton's second law of motion, $F=ma$. *Momentum* is defined by the multiplication of mass and velocity. According to Figure 3, we will have the rate of change of the u momentum within the patch equal to $\Delta x \Delta y \frac{\partial \rho u}{\partial t}$, the rates at which the u momentum enters through the left face is $\rho u^2 \Delta y$, the right face is $-\rho u^2 \Delta y$, the difference is $\frac{\partial \rho u^2}{\partial x} \Delta x \Delta y$. The upper face is $\rho u v \Delta x$ and the lower face is $-\rho u v \Delta x$, the difference is $\frac{\partial \rho u v}{\partial y} \Delta x \Delta y$. Then we have:

Pressure force acting on left face is $p \Delta y$

Pressure force acting on right face : $-(p \Delta y + \frac{\partial p}{\partial x} \Delta x \Delta y)$

Normal Viscous force acting on left face : $-\tau_{xx} \Delta y$

Normal Viscous force acting on right face : $\tau_{xx} \Delta y + \frac{\partial \tau_{xx}}{\partial x} \Delta x \Delta y$

Tangential viscous force on lower face is : $-\tau_{yx} \Delta x$

Figure 3: Control patch used to derive the Navier-Stokes equation

Tangential viscous force on upper face is : $\tau_{yx}\Delta x + \frac{\partial\tau_{yx}}{\partial x}\Delta x\Delta y$

Summing up all these contributions, we have

$$\frac{\partial\rho u}{\partial t}\Delta x\Delta y + \frac{\partial\rho u^2}{\partial x}\Delta x\Delta y + \frac{\partial\rho uv}{\partial y}\Delta x\Delta y = -\frac{\partial p}{\partial x}\Delta x\Delta y + \frac{\partial\tau_{xx}}{\partial x}\Delta x\Delta y + \frac{\partial\tau_{yx}}{\partial x}\Delta x\Delta y \quad (13)$$

Dividing each end by $\Delta x\Delta y$ and taking the limits as Δx and Δy going to zero, we will get the u direction momentum equation:

$$\frac{\partial\rho u}{\partial t} + \frac{\partial\rho u^2 + p}{\partial x} + \frac{\partial\rho uv}{\partial y} = \frac{\partial\tau_{xx}}{\partial x} + \frac{\partial\tau_{xy}}{\partial y} \quad (14)$$

taking the Stokes equations inside and considering the density value to be constant for incompressible fluid, we will have the following formula:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{1}{\rho}\frac{\partial p}{\partial x} = \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \quad (15)$$

where ν is $\frac{\mu}{\rho}$.

Extending the formula to 3D and putting the external force inside, eventually, we get the incompressible Navier-Stokes equation as follows:

$$\nabla \cdot u = 0 \quad (16)$$

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u - \frac{1}{\rho}\nabla p + \nu\nabla^2 u + f \quad (17)$$

where u is the velocity field, p is the pressure field, ν is the kinematics viscosity of the fluid, ρ is its density, f is an external force and ∇ is the vector of spatial partial derivatives. The first formula describes the conservation of mass. It indicates that the inflow and outflow of mass in a fluid element are balanced. The second formula is the conservation of momentum. It describes the velocity changes in time, due to convection, spatial variations in pressure and viscous forces.

In order to find the similarity between two different flows, we further define a dimensionless number, called the *Reynolds number*.

$$Re = VL/\nu \tag{18}$$

where $\nu = cm^2/sec$, $L = cm$, $V = cm/sec$. This parameter measures the relative importance of convection and viscosity. If the Reynolds number is relatively small, the fluid is viscous and the flow regime is laminar; if it is large, the fluid will be inviscid and the flow is turbulent. Basically, for a Reynolds number between 0 and 1000, the flow is smooth; for a Reynolds number between 2000 and 6000, some turbulence can be observed; for a Reynolds number between 10,000 and 10,000,000, extreme turbulence occurs. Solving the Navier-Stokes equation for different Reynolds numbers will get different kinds of fluid. If we rescale the parameters in formula 17, then we will have a new formula. Suppose $f=0$:

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u - \nabla p + \frac{1}{Re} \nabla^2 u \tag{19}$$

Many books and articles [13, 21, 41] have been published on how to solve this equation numerically. The difficulty in solving the Navier-Stokes equation is due to its nonlinear term. However, in several important settings, the Navier-Stokes equation will reduce to a set of linear PDEs. In the following paragraphs, we will first discuss the conditions and how to solve these linear PDEs, and then consider the nonlinear solutions.

Linear Partial Differential Equations

In fluid flow, there are two special cases: perfect flow and slow flow. *Perfect flow* is characterized by two properties: incompressibility and zero viscosity. We all know that viscosity will cause rotation in the flow. Then, if a flow has zero viscosity, it is free of rotation. *Slow flow* is an incompressible flow in which the viscous behavior of the flow dominates any inertial component of the flow. Both of these flows can be described by a set of linear PDEs. Weimer and Warren [48] proposed a method to use a subdivision scheme for vector fields to model flows, governed by linear PDEs. The system first defines a coarse initial vector field as a small set of vectors. The subdivision process then produces an arbitrarily dense discrete vector field which follows the initial field and satisfies the underlying partial differential equations.

Instead of solving the flow numerically, Wejchert and Haumann [49] used an analytic solution to the Navier Stokes equation. They simplified the fluid by making an assumption that it is inviscid, irrotational and incompressible and used simple flow primitives: uniform flow, sink, source and vortex to represent its movement. They also defined the models of particles and objects in flow. The movement of particles can be described by the Stokes drag equation. It gives the force exerted on a spherical particle with radius a , moving with relative velocity v^r in a fluid with viscosity η as in:

$$F = 6\pi a\eta v^r \tag{20}$$

The forces acting on the surface of an object can be resolved into the normal and tangential components with respect to the surface. The normal component of the force is due to the

pressure difference between the front and rear of the surface. The tangential force component is because of a fluid with viscosity moving across a surface.

$$F_n = \rho Av^2 \quad (21)$$

$$F_t = A\eta \frac{dv}{dy} \quad (22)$$

Kass and Miller [23] linearize the shadow water equations to simulate liquids. They reduce the Navier-Stokes equations down to solving for an evolving height field for the surface of a shallow body of liquid. The shortcoming of their method is that the simplifications do not capture the rotational motions of fluids.

Finite Difference Solution

The finite difference method is the most common approach used to solve the Navier-Stokes equation. It is based on the Taylor series to describe a derivative of variables as a difference of values in space and time. Then a differential variable $\frac{\partial F}{\partial x}$ will be changed to the new expression.

$$\frac{\partial F}{\partial x} = \frac{1}{2h}(F(x+h) - F(x-h)) + O(h^2) \quad (23)$$

Similarly, a second order derivative $\frac{\partial^2 F}{\partial x^2}$ will be written in the following formula.

$$\frac{\partial^2 F}{\partial x^2} = \frac{1}{h^2}(F(x+h) - 2F(x) + F(x-h)) + O(h^2) \quad (24)$$

Chen and Lobo [2, 4] solved a simplified Navier-Stokes equation in two dimensions using this approach. They mapped the pressure value to depth as the third dimension of a wave. This assumption works well for water because of the correlation between pressure and the depth of the water. The drawback of this approach is that they can only describe limited interaction between fluid and objects.

Based on Chen and Lobo's work, Foster and Metaxas [15, 16, 17] presented a solution of the full three dimensional Navier-Stokes equations to simulate incompressible and viscous fluid. They used a low resolution 3D volume to represent gas. The solid 3D objects in the scene are approximated as a series of regular voxels that are axially aligned to the coordinate used by the gas volume. It is illustrated in Figure 4(a). Together with an equation to describe the thermal buoyancy of the gas, and a formula to account for the convection and diffusion of the temperature field, they solved the NS equation in three dimensions. To guarantee the conservation of mass, they used iteration to calculate relaxation adjustment. However, their scheme suffered from the instability problem. For an explicit solver, there is always a restriction on the size of the time step. The method is stable only if the time step satisfies the condition $\Delta t < \Delta\tau/u$, where $\Delta\tau$ is the spacing of the computational grid. In other words, if the grid spacing is small or the velocity is large, the time step must be considerably small, or the iteration process will collapse. They also solved the problem of initial values and boundary conditions elegantly. Referring to Figure 4(c), they defined the velocity and temperature value at the boundary cell depending on the type of material or object that it represents. For example, a hot radiator can not allow gas to pass through it, thus, the velocity is set to zero for cell faces that represent the radiator boundary.

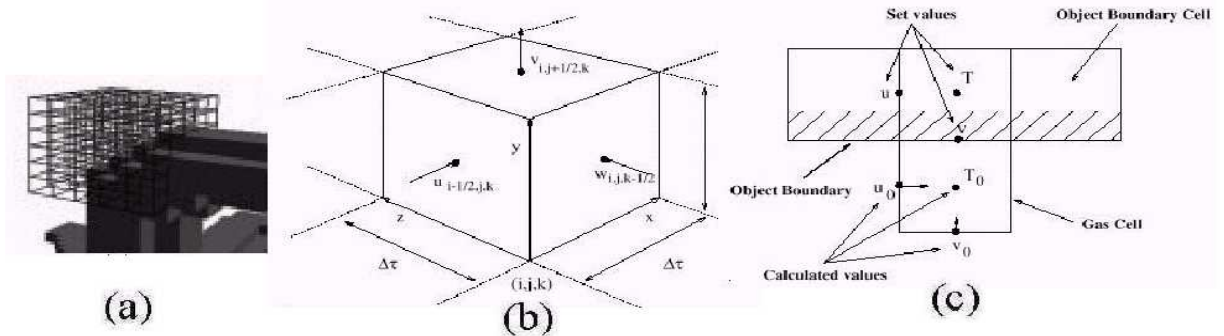


Figure 4: Image (a) shows the regular voxels used to approximate a scene and the medium around those objects. Image (b) shows the numbering convention for a single cell in the voxel grid. $\Delta\tau$ is the side length for each face of the cell. Image (c) describes the boundary between a gas and an object.

Tangentially, gas can flow freely along the surface, so velocity is set to the external tangential velocity. Temperature should flow freely from the radiator to the air, then, the temperature within the boundary cell is set to the designed temperature of the radiator.

Foster solved the 3D Navier-Stokes equation by making an assumption that the fluid is incompressible. What about the compressible fluid? In the film *The Prince of Egypt*, Witting from DreamWorks Feature Animation Studio [51] gave a compressible version of the 2D numerical solution. Another feature of their system is the use of images and animations as input devices, which allows animators to control initial conditions, source terms and movable internal boundaries in an easy and flexible way. They also implemented the fourth order Runge-Kutta scheme, which uses fourth order centered finite differencing for spatial derivatives on a regular grid with equal grid spacing. At boundary points and one point away, one-sided differencing is used. The fourth order accuracy may not be required for many systems, however, the effort in achieving this added accuracy is not significant, and the increased accuracy allows for the use of coarser grids to achieve high speed.

Yngve, et al. [53] also implemented a finite difference solution for compressible and viscous fluid. They used it to simulate the propagation of an explosion through the surrounding air. For their specific problem, because the steep pressure gradients are moving at supersonic speed, the explicit scheme diverges rapidly. They improved the integration technique by updating the Navier-Stokes equation and energy conservation equation in two steps: first using the non-convective terms and second using the convective derivatives. They also implemented the *donor-acceptor* method to solve the problem when mass, momentum, and energy are convected across steep pressure gradients. Although they used the same finite difference approach as Foster, their method does not have the time-consuming iteration part to conserve mass, since their model is a compressible fluid.

Finite Element Solution

The finite element method provides a greater flexibility in modeling complex geometries than the finite difference does. It has been widely used in solving structural, mechanical, heat transfer,

and fluid dynamics problems, as well as problems of other disciplines.

The general procedure of the method is that the whole volume is divided into smaller elements of finite dimensions called "*Finite Elements*". The original volume is then considered as an assembly of these elements. They are connected to each other at joints called "*Nodes*" or "*Nodal Points*" to form the entire structure. The properties of these individual elements are formulated, and from these the properties for the entire volume are obtained. Please refer to Bathe's book [1] for further details.

Implicit Solution

When an unknown variable can be found directly from known values of the variables, we call this method an *explicit scheme*. The formulas we discussed above are all explicit solvers. Now we will consider an implicit scheme. If during the discretization process, we get an equation where several unknown values are related to several known values, then it is called an *implicit scheme*. To generate the solution for the implicit scheme, we need to solve a set of simultaneous equations. It may appear that the implicit schemes require more computational effort, however, they are more stable than the explicit schemes. They are stable for all sizes of the time step.

Stam [44, 45] proposed an unconditionally stable model to solve the 3D Navier-Stokes equation. Similar to Foster, his work is also based on a low resolution grid, and it allows larger time steps to achieve faster animation. Before discussing his method in detail, let's consider two basic approaches to describing fluid behavior: the *Lagrangian* method and the *Eulerian* method. The *Eulerian* approach thinks of the fluid as a field of velocity, pressure or other state variables. It focuses on the fixed point in the fluid field and studies the relationship between the variable at one point and the adjunct points. However, the *Lagrangian* approach models the fluid as the movement of individual particles. The velocity and pressure are bonded with the moving particle. In one sentence, the fluid is flowing past Euler, who sits at a point and watches Lagrange go by. Because of the difference of the two methods, we have the following transport equation: for any function $f(x,t)$, the ordinary derivative $\partial/\partial t$ at points fixed in space (Euler) is related to the material derivative D/Dt at points moving with fluid velocity $v(x,t)$ (Lagrange).

$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + v \cdot \nabla f \quad (25)$$

The methods and formulas we discuss above are all based on the Euler scheme. The method used by Stam is derived from the Lagrangian scheme. He separates the Navier-Stokes equation into four parts: force, advection, diffusion and projection, each one with an implicit function. Inspired by the density equation solver, he makes use of the characteristics method of PDE to solve the most difficult nonlinear convection term. The method can be understood intuitively. At each time step, all the fluid particles are moved by the velocity of the fluid itself. Therefore, to obtain the velocity at a point x at the new time $t + \Delta t$, we backtrace the point x through the velocity field u over a time Δt . This defines a path $p(x,s)$ corresponding to a partial streamline of the velocity field. The new velocity at the point x is then set to the velocity that the particle, now at x , had at its previous location a time Δt ago. The problem with this approach is that it suffers from too much numerical dissipation; the simulated fluid tends to dampen more rapidly than actual fluid. This indicates it can't be used to simulate shocks and explosion animation.

Parallel Optimization

As the resolution of the volume representing the amorphous objects increase, and the need for modeling extremely turbulent fluids, we cannot achieve real time animation speed due to the CPU and memory limitation on a single-processor computer. This situation leads to the necessity of generating parallel algorithms to solve the Navier-Stokes equation. The parallel solution should make use of the 3D regular decomposition of the computational domain. Each processor should perform sequential calculations on local arrays; while communication between processors and global operations are implemented using Message Passing Interface routines. This is always a hot area in the study of the fluid dynamics field and there are many papers describing the parallel solutions of the Navier-Stokes equation in the AIAA(American Institute of Aeronautics and Astronautics), physics and Fluid Mechanics field.

0.3.3 Advection-Diffusion Equation

Advection-diffusion has been used to model the interaction between amorphous objects and the velocity field. They are employed for two reasons. First, they capture the main characteristics of many transport phenomena, such as when a non-reactive substance is injected into the fluid, it is advected by the fluid while diffusing at the same time. Second, they are simple enough to be understood by an animator with a limited knowledge of physics. Some common examples of this phenomena include the patterns created by milk stirred in coffee or the smoke rising from a cigarette. Examples of the non-reactive substance include the density of dust, smoke, cloud droplet and the temperature of a fluid. The advection-diffusion equation of the density distribution can be given by the following formula:

$$\frac{\partial \rho}{\partial t} = -u \nabla \rho + \kappa \nabla^2 \rho - \alpha \rho \quad (26)$$

The first term on the right hand side of the formula is the advection term that accounts for the effect of the velocity field on the density. The second term stands for the molecular diffusion at rate κ . The last term is the dissipation of density at rate α . Knowing the wind velocity, we can use a finite difference approach to solve the equation. Although it is linear in ρ , the computation cost increases rapidly in three dimension. Instead of considering points on grid, J. Stam [46] assume the density distribution is a weighted sum of a simple distribution f :

$$\rho(x, t) = \sum_{i=1}^n m_i(t) f(x - x_i(t), t - t_i) \quad (27)$$

In other words, the density distribution is a "fuzzy blobby" with time dependent field f , where $x_i(t)$ is the center of the mass, t_i is the time at which the "blob" ρ_i is created and $m_i(t)$ is the mass. If we suppose f is a gaussian distribution with a standard deviation σ_0 , then we can represent the convection of the blob by integrating its center of mass over the wind field, the diffusion of the blob by the convolution of the Gaussian functions and the dissipation by an exponential decay of the masses over time. Later, in 1995, Stam [47] improved his solution by reformulating the problem from a grid to "warped blobs". These blobs more accurately model the distortion that gases undergo when advected by wind fields. Another possible solution of the advection-diffusion equation is based on the implicit scheme suggested by Stam [44], which

we have discussed earlier. Because of the similarity between the AD equation and the Navier-Stokes equation, we can combine the implicit forms of the advection term and the diffusion term, and solve the formula in the same way as the velocity of fluid. Early in 1991, Witkin, Kass [50] and Turk [?] presented a method for texture synthesis based on reaction-diffusion, which had been proposed as a model of biological pattern formation. Similar to Advection-diffusion equation, the traditional reaction-diffusion system include terms of the diffusion and the dissipation. They implemented a finite difference approach and extended the conventional method by allowing anisotropic and spatially non-uniform diffusion, as well as multiple competing directions of diffusion.

0.3.4 Turbulence Theory

The aim of turbulence theory is to describe turbulence globally by using statistical methods instead of accurate fluid mechanics. The results obtained from this approach are limited and do not hold for accurate local prediction, however, the approach provides a plausible global solution avoiding extensive computation. Based on the theory, a turbulent stream may be regarded as the superposition of two motions: $u = \bar{U} + u'$, where \bar{U} is an average translative velocity, responsible for the global translative motion of the current, while u' is the additional random fluctuating motion. It can be regarded as the result of the simultaneous existence of turbulent perturbations (or *eddies*) of different sizes. Each of these eddies has a characteristic velocity of u_λ . The superposition of these eddies results in the chaotic movement that is characteristic of turbulent flow. An eddy is actually a local disturbance of a certain size and velocity within a turbulent field. Local disturbance means that an eddy is associated with a spatial location within the turbulent current.

Inspired by the spectral theory of turbulence, Sakas [39] employed the stochastic spectral synthesis method and defined turbulent motion in Fourier space to model clouds. Through his method, a cloud was constructed by overlapping sinusoidal waves with different amplitudes and random phases in a spatiotemporal frequency domain, then inverting the result to get a periodic space-time density field. A translation in Euclidean space can be achieved via a phase shift in the frequency domain.

There are two shortcomings in Sakas's method. One is called computational globality, because a single structure can't be calculated unless the complete field is generated. The other is called structural globality, because the Fourier spectrum represents only averages of all local structures; it cannot model individual structures, such as a single eddy. A consequence of this is that the generated texture does not interact with the environment naturally.

Based on turbulence theory, Stam [46] generated a multi-scale wind field model. Different from Sakas, he modeled gases as density distributions of particles. Although spectral synthesis is useful in generating turbulent wind fields, it is not ideal for directly generating density fields. Thus, he used an advection-diffusion equation to describe the evolution of the density distribution, which we have discussed earlier. His model employed simple wind field primitive to specify large scale wind behavior, and a three-dimensional random vector field varying over space and time to model the turbulent small-scale behavior.

Shinya and Fournier [42] introduced a simple stochastic wind field from the structural engineering field. The temporal characteristic of wind can be described as a low-pass in the frequency domain, where the cut-off frequency depends on the mean velocity of the wind field. Since the

spatio-temporal property of wind can be modeled by its co-spectra, they used FFT to synthesize the wind velocity. Because of the periodic property of the discrete Fourier transform, it is also convenient to model a large area and a long animation sequence.

0.4 Cellular Automata

0.4.1 Background

Cellular automata [40] were originally proposed by John Von Neumann as formal models of self-reproducing organisms. They are described as discrete dynamical systems. The space, time, and the states of the cells are discrete. Each cell in the regular spatial lattice can have any one of a finite number of states. The states of the cells in the lattice are updated according to a local rule. That is, the state of a cell at a given time depends only on its own state one time step previously, and the states of its nearby neighbors at the previous time step. All cells on the lattice are updated synchronously. Thus, the state of the entire lattice advances in discrete time steps. Cellular automata may produce extremely complex structures from the evolution of rather simple local rules. One of the famous examples is Conway's "life" where, on a 2D grid, cells are either dead or alive. In one time step, the cell's fate is dependent on the number of living neighboring cells. If that number exceeds or falls short of given limits, then the central cell dies; otherwise it survives or a new cell is generated. By carefully selecting initial configurations, we can have many interesting patterns.

Nagel [31] extremely simplified the cloud dynamics based on cellular automata. In his model, each of the cells has only three variables: humidity, phase transition and existence of cloud, represented by 0 or 1. The cloud evolution is calculated by easy Boolean operations of variables on neighboring cells. Based on their work, Dobashi [9] implemented a fast animation of clouds over a mountain. They used probability to control the cloud growth simulation rate, cloud extinction rate and the effect of advection by the wind. This method can generate a visually convincing, however, not physically exact cloud animation. It is fast and uses only a small amount of memory and is suitable for modeling cumulus types of clouds.

0.4.2 Lattice Gas Model

Hydrodynamics was one of the early successes of the theory of cellular automata, and it remains one of the areas that is best developed and most important for practical applications. As we all know, fluid behavior is a similar self-organizing process evolving from the microscopic collisions of atoms or molecules. We can use cellular automata to simulate the microscopic movements in order to get the continuum macroscopic equations of fluid dynamics in two and three dimensions. The class of cellular automata used for the simulation of fluid dynamics is called the "*lattice gas model*" [10]. The main difference between the LGA and the traditional numerical approaches of the NS equations is that in the LGA the physical evolution rules are discrete, while in the latter, the discretization is performed on the level of the macroscopic flow equations.

HPP Model

The first LGA was called the HPP model on a square lattice [19]. Particles of unit mass and unit speed are moving along the lattice links. Not more than one particle in a given direction can be

found at a given time and node. When two particles arrive at a node from opposite directions, they immediately leave the node in the two other, previously unoccupied direction. These rules obviously conserve mass (particle number) and momentum. The main problem with this model is that the gaseous behavior it modeled is not isotropic.

FHP Model

The historically important lattice gas model was FHP model (Figure 5), introduced by Frisch, Hasslacher and Pomeau [19] in 1986.

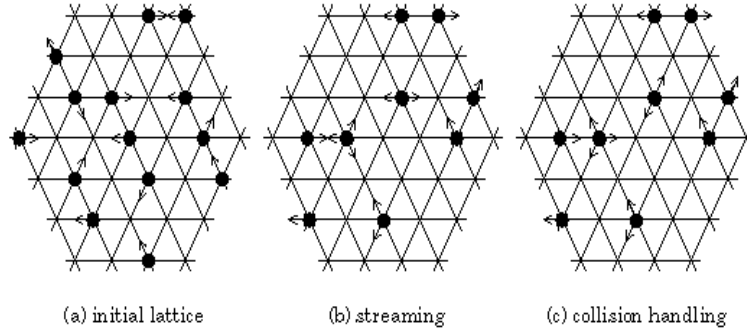


Figure 5: Update mechanism of an FHP model. Arrows denote a particle and its moving direction. In Figures a to c the propagation (streaming) and collision phases are shown for some initial configuration.

It is a two dimensional triangular lattice used to ensure macroscopic isotropy. In this model, each cell has six nearest neighbors and consequently six possible velocity directions. Updating still involves two types of rules: propagation and collision. Propagation means the particle will move to the nearest neighbor along its velocity direction. Collision is the most important part, particles may change directions. It is decided by the collision operator. No matter how we define the collision operator, the conservation of mass and momentum must be satisfied. For instance, when two particles enter the same node with opposite velocities, both of them are deflected by 60 degrees, such that the net momentum in the post collision state remains zero, and when multiple states are possible, a random selection is made. Figure 6 represents the collision rules for the FHP model.

It can be demonstrated that by observing these microscopic rules, we can simulate the following macroscopic equations:

$$\frac{\partial u}{\partial t} = -(g(\rho)u \cdot \nabla)u - \frac{1}{\rho} \nabla p + \nu \nabla^2 u \quad (28)$$

where ρ is the density and can be calculated as the average number of particles per cell. If we renormalize the time t , the viscosity ν and the pressure $p(t' = g(\rho)t, \nu' = \frac{\nu}{g(\rho)}, p' = \frac{p}{g(\rho)})$, the Navier-Stokes equation for incompressible fluids without external force can be recovered.

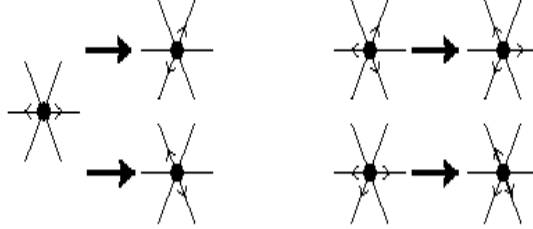


Figure 6: Collision rules of the FHP model. An arrow denotes a particle and its moving direction. When multiple post collision states are possible, a random selection is made. The collision rules satisfy mass and momentum conservation.

Face-centered Hypercube Model

Because the difficulty of satisfying the isotropy requirement for the three dimensional model, Frisch, et al. [18] suggested using the 3D projection of the four-dimensional *face-centered hypercube* (FCHC). This lattice consists of 24 nodes $((\pm 1, \pm 1, 0, 0), (\pm 1, 0, \pm 1, 0), (\pm 1, 0, 0, \pm 1), (0, \pm 1, \pm 1, 0), (0, \pm 1, 0, \pm 1), (0, 0, \pm 1, \pm 1))$. Each cell has 24 neighbors and therefore the number of possible collisions is 2^{24} , which is much higher than the 2^6 of the 2D FHP model.

All in all, the main idea of LGA is to model fluid dynamics by a particle propagation and collision process on a regular lattice. It has some unique advantages with respect to standard numerical solutions:

1. boundary conditions are easy to implement and the method is especially suitable for simulating fluid flow in complex geometries;
2. the model is ideal for massively parallel computing because the updating of a node involves only its nearest neighbors;
3. the method is stable compared to numerical solutions

Although LGA has proven to be very useful for modeling hydrodynamics, one major drawback of the method is the statistical noise in the computed hydrodynamic fields. This is a direct consequence of the single particle Boolean operation. One method of smoothing is to average in space and time. In practice, spatial averages are taken over 64, 256, 512 or 1024 neighboring cells for time-dependent flow in two dimensions. This dramatically limits the space resolution we can get. LGA can only model a limited range of flow velocity. This problem is inherent in a model that assumes a single speed for all particles. Another problem is when we use the FCHC model to simulate 3D flow. The memory and speed cost of calculating a collision step is too high.

0.4.3 Lattice Boltzman Model

Based on the shortcoming of the LGA model, if we trace a density of particles along each bond of the lattice, following the motion of an average number of particles, the noise in the model can be eliminated using the *Lattice Boltzman model (LBM)* [6]. LBM as an independent numerical

method for hydrodynamics simulation was introduced by McNamara and Zanetti in 1988. The discrete particles at each grid point are replaced by the averaged particle distribution f_{qi} . f_{qi} can be any real values. The indices qi describe the D-dimensional sublattices defined by the permutations of $(\pm 1, \dots, \pm 1, 0, \dots, 0)$ where q is the number of non-zero component and i counts the sub-lattice vectors. And in contrast to the LGA models, particles with zero velocity are allowed. Similar to the Lattice gas model, in LBM space, the time and particle packet velocities are also discretized. The time evolution of the system is governed by the Boltzman equation.

$$f_{qi}(\vec{r} + \vec{c}_{qi}, t + 1) - f_{qi}(\vec{r}, t) = \Omega_{qi} \quad (29)$$

the term \vec{c}_{qi} represents the packet velocity along the lattice links; Ω is a general collision operator. The collisions are again completely local, so that the LBM is, and likely the LGA, very efficiently parallelizable. The macroscopic variables density and velocity follow from the respective velocity moments of the particle distributions.

$$\rho = \sum_{qi} f_{qi} \quad (30)$$

$$\vec{v} = \frac{1}{\rho} \sum_{qi} f_{qi} \vec{c}_{qi} \quad (31)$$

Now the most important part is to how to decide the collision operator. Supposing there is always a local equilibrium particle distribution f_{qi}^{eq} dependent on the conserved quantities ρ and \vec{v} only. Then we can change the Boltzman equation to

$$f_{qi}(\vec{r} + \vec{c}_{qi}, t + 1) - f_{qi}(\vec{r}, t) = \frac{1}{\tau} (f_{qi}(\vec{r}, t) - f_{qi}^{eq}(\rho, \vec{v})) \quad (32)$$

$$f_{qi}^{eq}(\rho, \vec{v}) = \rho(A_q + B_q(\vec{c}_{qi} \cdot \vec{v}) + C_q(\vec{v})^2 + D_q(\vec{c}_{qi} \cdot \vec{v})^2 + \dots) \quad (33)$$

The term τ is the relaxation time scale. The coefficients A_q to D_q are dependent on the employed lattice geometry. For the models we are discussing, they are constant. For the sake of computational efficiency, one tries to find the minimum necessary number of particle distributions. In order to simulate an amorphous object in 3D, we can use “D3Q15” lattice geometry. The “D3Q15” lattice consists of 15 nodes on three sub-lattices:

1. $q=0$, the cell $(0,0,0)$ has particles with zero velocity;
2. $q=1$, the sixth nearest neighbors $(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)$ where the particles move with unit velocity
3. $q=3$, the eight third-nearest neighbors $(\pm 1, \pm 1, \pm 1)$ where the particles move with a velocity of $\sqrt{3}$.

The following Figure 7 is the geometric structure of the ”D3Q15” model.

Initial conditions are usually specified in terms of densities and velocities. For the Lattice Boltzmann method, these macroscopic values have to be translated into corresponding microscopic particle packets for each fluid phase. Boundary conditions can also be set easily. Boundary nodes are placed half-way between the grid points. Then the propagation step is modified according to the BBL rule, which make particle packets bounce back along the link. Since the Lattice Boltzmann method includes both integer and floating point operations, it potentially suffers from numerical instability.

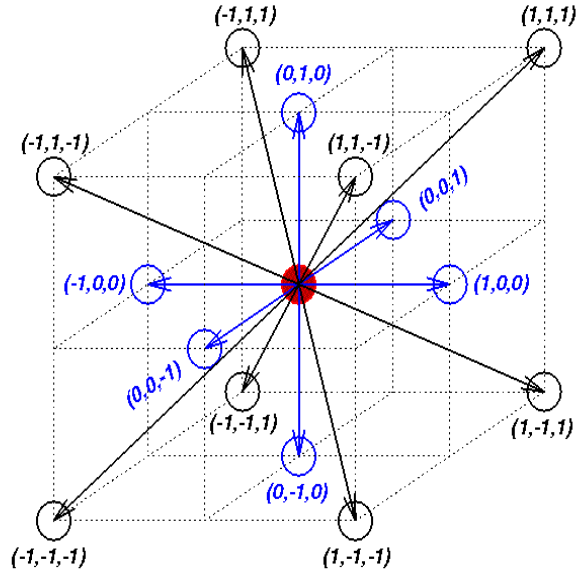


Figure 7: Sketch of the D3Q15 lattice geometry. The velocity directions of the 14 moving particle distributions are shown as arrows. In each time step the particle packets move along the predefined directions and then relax toward the equilibrium distribution. This relaxation rate determines the physical viscosity.

0.4.4 Parallel Optimization

The LGM is an ideal candidate for parallel execution since most of the calculations are done locally [22, 29]. The basic idea behind parallelization is to distribute a global problem among a number of separate processors. In parallelization, the concept of "Single Instruction Multiple Data" means that the same program is executed by all processors in parallel, and each of them works on its individual piece of data. This idea is suited for the problem of LGM. Munders implemented a parallel approach for the LBM based on a CM-5 parallel computer.

0.5 Conclusion

Amorphous objects modeling and manipulation are essential to achieving realism in virtual environments. However, since they don't have well-defined surfaces and boundaries, their intensity values, structures may change with time, and the difficulty of correctly modeling the interaction of amorphous objects with the surrounding environment is always a hard problem needing to be solved. In this report, we surveyed literature in visualization, graphics and fluid mechanics. We have discussed the method of procedural modeling, particle system modeling, physical-based modeling, lattice gas model, lattice Boltzman model and their applications in detail. In the following table, we illustrated the advantages and disadvantages of each of the methods:

	Procedural model	Particle system	Physical-based model		Lattice Gas	Lattice Boltzman
			explicit	implicit		
Variety	cloud, steam fire, dust	all	all	all	all	all
Dimensions	2D,3D	2D,3D	2D,3D	2D,3D	2D,3D	2D,3D
Speed	fast	fast	slow	slow	fast	fast
Parallel	yes	yes	yes	yes	yes	yes
Interaction	bad	simple kinematic rules	good	good	good	good
Stability	good	good	limited size of timestep	good	statistical noise	numerical instability
User Interface	noise function parameters	simple	simple	simple	simple	set particle distribution

Bibliography

- [1] K. J. Bathe. *Finite Element Procedures*. Prentice-Hall, Englewood Cliffs, NJ, 1996.
- [2] J. X. Chen, N. da V. Lobo, C. E. Hughes, and J. M. Moshell. Real-time fluid simulation in a dynamic virtual environment. *IEEE Computer Graphics and Applications*, pages 52–61, May 1997.
- [3] J. X. Chen, X. Fu, and E. J. Wegman. Real-time simulation of dust behavior generated by a fast traveling vehicle. *ACM Transactions on Modeling and Computer Simulation*, 9(2):81–104, April 1999.
- [4] J. X. Chen and N. DA V. Lobo. Toward interactive-rate simulation of fluids with moving using navier-stokes equationsn. *Graphical Models and Image Processing*, 57(2):107–116, March 1995.
- [5] M. Chen, A. E. Kaufman, and R. Yagel. *Volume Graphics*. Springer, 2000.
- [6] S. Chen and G. D. Doolean. Lattice boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.*, 30:329–364, 1998.
- [7] R. A. Crawfis and N. Max. Texture splats for 3d scalar and vector field visualization. *Proceeding of IEEE Visualization*, October 1993.
- [8] W. E. Carlson D. S. Ebert and R. E. Parent. Solid spaces and inverse particle systems for controlling the animation of gases and fluids. *The Visual Computer*, 10:471–483, 1994.
- [9] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, and T. Nishita. A simple, efficient method for realistic animation of clouds. *Proceedings of SIGGRAPH 2000*, pages 121–128, August 2000.
- [10] G. D. Doolen. *Lattice Gas Methods for Partial Differential Equations*. Addison-Wesley Publishing Company, 1990.
- [11] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling: A Procedural Approach*. Academic Press, Inc, 1994.
- [12] D. S. Ebert and R. E. Parent. Rendering and animation of gaseous phenomena by combining fast volume and scanline a-buffer techniques. *Computer Graphics*, 24(4):357–366, 1990.
- [13] S. J. Farlow. *Partial Differential Equations for Scientists and Engineers*. Dover Publications, Inc. New York, 1982.

- [14] J. A. Fay. *Introduction to Fluid Mechanics*. MIT Press, 1994.
- [15] N. Foster and D. Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.
- [16] N. Foster and D. Metaxas. Controlling fluid animation. *Proceeding of CGI*, pages 178–188, 1997.
- [17] N. Foster and D. Metaxas. Modeling the motion of a hot, turbulent gas. *Proceedings of SIGGRAPH'97*, pages 181–188, August 1997.
- [18] U. Frisch, D. d’Humières, B. Hasslacher, Y. Pomeau, and J. Rivet. Lattice gas hydrodynamics in two and three dimensions. *Complex Systems*, 1:649–707, 1987.
- [19] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the navier-stokes equations. *Physical Review Letters*, 56(14):1505–1508, April 1986.
- [20] G. Gardner. Visual simulation of clouds. *Proceedings of SIGGRAPH'85*, 19(3):297–303, July 1985.
- [21] N. Gershenfeld. *The Nature of Mathematical Modeling*. Cambridge University Press, 1999.
- [22] B. D. Kandhai. *Large Scale Lattice-Boltzmann Simulations*. PhD thesis, University of Amsterdam, December 1999.
- [23] M. Kass and G. Miller. Rapid, stable fluid dynamics for computer graphics. *Proceeding of SIGGRAPH*, 24(4):49–57, August 1990.
- [24] A. E. Kaufman. *Volume Visualization Tutorial*. IEEE Computer Society Press, Los Alamitos, CA, 1991.
- [25] A. E. Kaufman. Volume graphics. *IEEE Computer*, 26(7):51–64, July 1993.
- [26] A. E. Kaufman. Volume visualization. *ACM Computing Surveys*, 28(1):165–167, March 1996.
- [27] S. A. King, R. A. Crawfis, and W. Reid. *Fast Volume Rendering and Animation of Amorphous Phenomena*, pages 229–242. 2000.
- [28] N. Max and R. Crawfis. Visualizing wind velocities by advecting cloud textures. *Proceeding of Visualization*, pages 179–183, October 1992.
- [29] D. Muders. *Three-Dimensional Parallel Lattice Boltzmann Hydrodynamics Simulations of Turbulent Flows in Interstellar Dark Clouds*. PhD thesis, University at Bonn, August 1995.
- [30] F. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrains. *Proceedings of SIGGRAPH'89*, 20(3):41–50, July 1989.
- [31] K. Nagel and E. Raschke. Self-organizing criticality in cloud formation. *Physica A*, 182:519–531, 1992.

- [32] Paramount. Genesis demo from star trak ii: The wrath of khan. *SIGGRAPH Video Review*, (11), June 1982.
- [33] K. Perlin. An image synthesizer. *Proceedings of SIGGRAPH'85*, 19(3):287–296, July 1985.
- [34] K. Perlin and E. M. Hoffert. Hypertexture. *Proceedings of SIGGRAPH'89*, 20(3):253–262, July 1989.
- [35] W. T. Reeves. Particle system—a technique for modeling a class of fuzzy objects. *Proceeding of SIGGRAPH*, 17(3):359–376, July 1983.
- [36] W. T. Reeves and R. Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Proceedings of SIGGRAPH'85*, 19(3):22–26, July 1985.
- [37] M. Roth and R. Guritz. Case study: Visualization of volcanic ash clouds. *IEEE Visualization*, pages 386–390, 1994.
- [38] H. E. Rushmeier, A. Hamins, and M. Y. Choi. Case study: Volume rendering of pool fire data. *IEEE Visualization*, pages 382–385, 1994.
- [39] G. Sakas. Modeling and animating turbulent gaseous phenomena using spectral synthesis. *The Visual Computer*, pages 200–212, 1993.
- [40] P. Sarkar. A brief history of cellular automata. *ACM Computing Surveys*, 32(1):80–107, March 2000.
- [41] C. T. Shaw. *Using Computational Fluid Dynamics*. Prentice Hall, 1992.
- [42] M. Shinya and A. Fournier. Stochastic motion-motion under the influence of wind. *Proceeding of Eurographics*, 11:119–128, 1992.
- [43] K. Sims. Particle animation and rendering using data parallel computation. *Proceeding of SIGGRAPH*, 24(4):405–413, August 1990.
- [44] J. Stam. Stable fluids. *Proceedings of SIGGRAPH'99*, pages 121–128, August 1999.
- [45] J. Stam. Interacting with smoke and fire in real time. *Communications of the ACM*, 43(7):76–83, 2000.
- [46] J. Stam and E. Fiume. Turbulent wind fields for gaseous phenomena. *Proceedings of SIGGRAPH'93*, pages 369–376, August 1993.
- [47] J. Stam and E. Fiume. Depiction of fire and other gaseous phenomena using diffusion processes. *Proceedings of SIGGRAPH'95*, pages 129–136, August 1995.
- [48] H. Weimer and J. Warren. Subdivision schemes for fluid flow. *Proceedings of SIGGRAPH'99*, pages 111–120, August 1999.
- [49] J. Wejchert and D. Haumann. Animation aerodynamics. *Proceeding of SIGGRAPH*, 25(4):19–22, July 1991.

- [50] A. Witkin and M. Kass. Reaction-diffusion textures. *Proceeding of SIGGRAPH*, 25(4):299–308, July 1991.
- [51] P. Witting. Computational fluid dynamics in a traditional animation environment. *Proceedings of SIGGRAPH'99*, pages 129–136, August 1999.
- [52] L. Yaegel and C. Upson. Combining physical and visual simulation-creation of the planet jupiter for the film "2010". *Proceeding of SIGGRAPH*, 20(4):85–93, August 1986.
- [53] G. D. Yngve, J. F. O'Brien, and J. K. Hodgins. Animating explosions. *Proceedings of SIGGRAPH 2000*, pages 121–128, August 2000.